

# STOCHASTIC QUASI-NEWTON METHODS FOR NONCONVEX STOCHASTIC OPTIMIZATION

XIAO WANG <sup>\*</sup>, SHIQIAN MA <sup>†</sup>, AND WEI LIU <sup>‡</sup>

December 1, 2014

**Abstract.** In this paper we study stochastic quasi-Newton methods for nonconvex stochastic optimization, where we assume that only stochastic information of the gradients of the objective function is available via a stochastic first-order oracle ( $\mathcal{SFO}$ ). Firstly, we propose a general framework of stochastic quasi-Newton methods for solving nonconvex stochastic optimization. The proposed framework extends the classic quasi-Newton methods working in deterministic settings to stochastic settings, and we prove its almost sure convergence to stationary points. Secondly, we propose a general framework for a class of randomized stochastic quasi-Newton methods, in which the number of iterations conducted by the algorithm is a random variable. The worst-case  $\mathcal{SFO}$ -calls complexities of this class of methods are analyzed. Thirdly, we present two specific methods that fall into this framework, namely stochastic damped-BFGS method and stochastic cyclic Barzilai-Borwein method. Finally, we report numerical results to demonstrate the efficiency of the proposed methods.

**Keywords:** Stochastic Optimization, Nonconvex Optimization, Stochastic Approximation, Quasi-Newton Method, BFGS Method, Barzilai-Borwein Method, Complexity

**Mathematics Subject Classification 2010:** 90C15; 90C30; 62L20; 90C60

**1. Introduction.** In this paper, we consider the following stochastic optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) \quad (1.1)$$

where  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is continuously differentiable and possibly nonconvex. We assume that the exact information of function values and gradients of  $f$  are not available and only noisy gradients of  $f$  can be obtained via subsequent calls to a *stochastic first-order oracle* ( $\mathcal{SFO}$ ). Problem (1.1) arises in many applications, including machine learning [27], simulation-based optimization [13], and mixed logit modeling problems in economics and transportation [4, 2, 21]. In these applications, the objective function is sometimes given in the form of an expectation of certain function with a random variable being a parameter:

$$f(x) = \mathbb{E}[F(x, \xi)], \quad \text{or} \quad f(x) = \int_{\Xi} F(x, \xi) dP(\xi),$$

where  $\xi$  denotes a random variable and its distribution  $P$  is supported on  $\Xi$ . Since in many cases either the integral is difficult to evaluate or function  $F(\cdot, \xi)$  is not given explicitly, the function values and gradients of  $f$  cannot be easily obtained and only noisy gradient information of  $f$  is available.

The idea of employing stochastic approximation (SA) to solve stochastic programming problems can be traced back to the seminal work by Robbins and Monro [38]. The classical SA method mimics the steepest

---

<sup>\*</sup>School of Mathematical Sciences, University of Chinese Academy of Sciences, China. Email: wangxiao@ucas.ac.cn. Research of this author was supported in part by Postdoc Grant 119103S175, UCAS President Grant Y35101AY00 and NSFC Grant 11301505.

<sup>†</sup>Corresponding author. Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N. T., Hong Kong, China. Email: sqma@se.cuhk.edu.hk. Research of this author was supported in part by the Hong Kong Research Grants Council General Research Fund (Grant 14205314).

<sup>‡</sup>IBM T. J. Watson Research Center, Yorktown Heights, New York, USA. Email: weiliu@us.ibm.com

gradient descent method using a stochastic gradient, i.e., it updates the iterates via

$$x_{k+1} = x_k - \alpha_k G_k,$$

where  $G_k$  is an unbiased estimate of the gradient of  $f$  at  $x_k$ , and  $\alpha_k$  is a stepsize for the stochastic gradient step. In the literature, the SA method is also referred to as stochastic gradient descent (SGD) method. The SA method has been further studied extensively in [7, 12, 14, 36, 37, 40, 41], and the main focus in these papers has been the convergence of SA in different settings. Recently, there have been lots of interests in analyzing the worst-case complexity of SA methods. Works along this direction were mainly ignited by the complexity theory developed by Nesterov for first-order methods engaging in solving convex optimization problems [32, 33]. Nemirovski *et al.* [31] proposed a mirror descent SA method for solving nonsmooth convex stochastic programming problem  $x^* := \operatorname{argmin}\{f(x) \mid x \in X\}$  and analyzed its worst-case iteration complexity, where  $f$  is nonsmooth and convex and  $X$  is a convex set. Specifically, it was shown in [31] that for any given  $\epsilon > 0$ , the proposed mirror descent SA method needs  $O(\epsilon^{-2})$  iterations to obtain an  $\bar{x}$  such that  $\mathbb{E}[f(\bar{x}) - f(x^*)] \leq \epsilon$ , where  $\mathbb{E}[y]$  denotes the expectation of the random variable  $y$ . Other SA methods with provable complexity analysis for solving convex stochastic optimization problems have also been studied in [15, 22, 23, 24, 25].

It is noted that the SA methods mentioned above all concentrated on convex stochastic optimization problems. Recently there have been lots of interests on SA methods for nonconvex stochastic optimization problems (1.1) in which  $f$  is a nonconvex function. Ghadimi and Lan [17] proposed a randomized stochastic gradient (RSG) method for nonconvex stochastic optimization (1.1). RSG returns an iterate from a randomly chosen iteration as an approximate solution. It is shown in [17] that to return an  $\epsilon$ -solution  $\bar{x}$ , i.e.,  $\mathbb{E}[\|\nabla f(\bar{x})\|^2] \leq \epsilon$ , the total number of  $\mathcal{SFO}$ -calls needed by RSG is in the order of  $O(\epsilon^{-2})$ . Ghadimi and Lan [16] also studied an accelerated stochastic SA method for solving stochastic optimization problems (1.1) based on Nesterov's accelerated gradient method, which improved the complexity for convex cases from  $O(\epsilon^{-2})$  to  $O(\epsilon^{-4/3})$ . A class of nonconvex stochastic optimization problems, in which the objective function is a composition of a nonconvex function  $f$  and a convex nonsmooth function  $g$ , i.e.,  $x^* := \operatorname{argmin}\{f(x) + g(x) : x \in \mathbb{R}^n\}$ , was considered by Ghadimi *et al.* in [19], and a mini-batch SA method was proposed and its worst-case  $\mathcal{SFO}$ -calls complexity was analyzed. In [9], a stochastic block mirror descent method, which incorporates the block-coordinate decomposition scheme into stochastic mirror-descent methodology, was proposed for a nonconvex stochastic optimization problem  $x^* = \operatorname{argmin}\{f(x) : x \in X\}$  with  $X$  having a block structure. More recently, Wang *et al.* [44] proposed a penalty method for nonconvex stochastic optimization problems with nonlinear constraints, and also analyzed its  $\mathcal{SFO}$ -calls complexity.

The aforementioned methods are all first-order methods in the sense that they only use (stochastic) first-order derivative information of the objective function. In this paper, we consider methods for solving (1.1) that employ certain approximate second-order derivative information of the objective function. Since approximate second-order information is used, this kind of methods are expected to take less number of iterations to converge, with the price that the per-iteration computational effort is possibly increased. Along this line, there have been some works in designing stochastic quasi-Newton methods for unconstrained stochastic optimization problems. Methods of this type usually employ the following updates

$$x_{k+1} = x_k - \alpha_k B_k^{-1} G_k, \quad \text{or} \quad x_{k+1} = x_k - \alpha_k H_k G_k, \quad (1.2)$$

where  $B_k$  (resp.  $H_k$ ) is a positive definite matrix that approximates the Hessian matrix (resp. inverse of

the Hessian matrix) of  $f(x)$  at  $x_k$ . Some representative works in this class of methods are discussed in the following. Among the various SGD methods, the adaptive subgradient (AdaGrad) proposed in [10] has been proven to be quite efficient in practice. AdaGrad takes the form of (1.2) with  $B_k$  being a diagonal matrix that estimates the diagonal of the squared root of the uncentered covariance matrix of the gradients. [3] also studied the method using SGD with a diagonal rescaling matrix based on the secant condition associated with quasi-Newton methods. In addition, it was shown in [3] that if  $B_k$  is chosen as the exact Hessian at the optimal solution  $x^*$ , the number of iterations needed to achieve an  $\epsilon$ -solution  $\bar{x}$ , i.e.,  $\mathbb{E}[f(\bar{x}) - f(x^*)] \leq \epsilon$ , is in the order of  $O(\epsilon^{-1})$ . However, the optimal solution of the problem usually cannot be obtained beforehand, so the exact Hessian information remains unknown. [39] discussed the necessity of including both Hessian and covariance matrix information in a (stochastic) Newton type method. The quasi-Newton method proposed in [5] uses some subsampled Hessian algorithms via the sample average approximation (SAA) approach to estimate Hessian-vector multiplications. In [6], the authors proposed to use the SA approach instead of SAA to estimate the curvature information. This stochastic quasi-Newton method is based on L-BFGS [26] and performs very well in some problems arising from machine learning, but no theoretical convergence analysis was provided in [6]. Stochastic quasi-Newton methods based on BFGS and L-BFGS updates were also studied for online convex optimization in Schraudolph *et al.* [42], with no convergence analysis provided, either. Mokhtari and Ribeiro [29] propose a regularized stochastic BFGS method (RES) for solving (1.1) with  $f$  being strongly convex, and proved its almost sure convergence. Recently, Mokhtari and Ribeiro [30] proposed an online L-BFGS method that is suitable for strongly convex stochastic optimization problems arising in the regime of large scale machine learning, and analyzed its global convergence. It should be noted that all the aforementioned methods based on stochastic quasi-Newton information mainly focused on solving convex or strongly convex stochastic optimization problems.

As discovered by several groups of researchers [3, 29, 42], when solving convex stochastic optimization problems, stochastic quasi-Newton methods may result in nearly singular Hessian approximations  $B_k$  due to the presence of stochastic information. [29] proposed a regularized BFGS update strategy which can preserve the positive-definiteness of  $B_k$ . However, for nonconvex optimization problems, preserving the positive-definiteness of  $B_k$  is difficult even in deterministic settings. In classic quasi-Newton methods for nonconvex deterministic optimization, line search techniques are usually incorporated to guarantee the positive-definiteness of  $B_k$ . However, performing the line search techniques in stochastic optimization is no longer practical, because the exact function values are not available. Therefore, a crucial issue in applying quasi-Newton methods to solve nonconvex stochastic optimization (1.1) is how to keep the positive-definiteness of the updates  $B_k$  without using the line search techniques. In this paper, we will discuss and address this issue. Our contributions in this paper are as follows.

1. We propose a general framework of stochastic quasi-Newton methods for solving nonconvex stochastic optimization problem (1.1). In addition, we analyze its almost sure convergence to the stationary point of (1.1).
2. We propose a general framework of randomized stochastic quasi-Newton methods for solving (1.1). In this kind of methods, the methods return an iterate from a randomly chosen iteration. We analyze their worst-case  $\mathcal{SFO}$ -calls complexity to find an  $\epsilon$ -solution  $\bar{x}$ , i.e.,  $\mathbb{E}[\|\nabla f(\bar{x})\|^2] \leq \epsilon$ .
3. We propose two concrete stochastic quasi-Newton update strategies, namely stochastic damped-BFGS update and stochastic cyclic-BB-like update, to adaptively generate positive definite Hessian approximations. Both strategies fit into the proposed general frameworks, so the established convergence and complexity results apply directly.

**Notation.** The gradient of  $f(x)$  is denoted as  $\nabla f(x)$ . The subscript  $k$  refers to the iteration number in an algorithm, e.g.,  $x_k$  is the  $k$ -th  $x$  iterate. Without specification,  $\|x\|$  represents the Euclidean norm of vector  $x$ . Both  $\langle x, y \rangle$  and  $x^\top y$  with  $x, y \in \mathbb{R}^n$  denote the Euclidean inner product of  $x$  and  $y$ .  $\lambda_{\max}(A)$  denotes the largest eigenvalue of a symmetric matrix  $A$ .  $A \succeq B$  with  $A, B \in \mathbb{R}^{n \times n}$  means that  $A - B$  is positive semidefinite. In addition,  $\text{mod}(a, b)$  with two positive integers  $a$  and  $b$  denotes the modulus of division  $a/b$ . We also denote by  $P_\Omega$  the projection onto a closed convex set  $\Omega$ .

**Organization.** The rest of this paper is organized as follows. In Section 2, we present a general framework of stochastic quasi-Newton methods for nonconvex stochastic optimization (1.1) and analyze its convergence in expectation. In Section 3, we present a general framework of randomized stochastic quasi-Newton methods and analyze its worst-case  $\mathcal{SFO}$ -calls complexity for returning an  $\epsilon$ -solution. In Section 4, we propose two concrete quasi-Newton update strategies, namely stochastic damped-BFGS update and stochastic cyclic-BB-like update. In Section 5 we report some numerical experimental results. Finally, we draw our conclusions in Section 6.

**2. A general framework for nonconvex stochastic quasi-Newton methods.** In this section we study the stochastic quasi-Newton methods for nonconvex stochastic optimization problem (1.1). We assume that only noisy gradient information of  $f$  is available via  $\mathcal{SFO}$  calls. Namely, for the input  $x$ ,  $\mathcal{SFO}$  will output a stochastic gradient  $G(x, \xi)$  of  $f$ , where  $\xi$  is a random variable whose distribution is supported on  $\Xi \subseteq \mathbb{R}^d$ . Here we assume that  $\Xi$  does not depend on  $x$ .

We now give some assumptions required throughout this paper.

**AS.1**  $f \in \mathcal{C}^1(\mathbb{R}^n)$ , i.e.,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is continuously differentiable.  $f(x)$  is lower bounded by  $f^{low}$  for any  $x \in \mathbb{R}^n$ .  $\nabla f$  is globally Lipschitz continuous with Lipschitz constant  $L$ .

**AS.2** For any iteration  $k$ , we have

$$a) \quad \mathbb{E}_{\xi_k} [G(x_k, \xi_k)] = \nabla f(x_k), \quad (2.1)$$

$$b) \quad \mathbb{E}_{\xi_k} [\|G(x_k, \xi_k) - \nabla f(x_k)\|^2] \leq \sigma^2, \quad (2.2)$$

where  $\sigma > 0$  is the noise level of the gradient estimation, and  $\xi_k, k = 1, \dots$ , are independent to each other, and they are also assumed to be independent of  $x_k$ .

In SGD methods, iterates are normally updated through

$$x_{k+1} = x_k - \alpha_k G(x_k, \xi_k), \quad (2.3)$$

or the following mini-batch version

$$x_{k+1} = x_k - \alpha_k \cdot \frac{1}{m_k} \sum_{i=1}^{m_k} G(x_k, \xi_{k,i}), \quad (2.4)$$

where  $m_k \geq 1$  is a positive integer and refers to the *batch size* in the  $k$ -th iteration. For deterministic unconstrained optimization, quasi-Newton methods have been proven to perform better convergence speed than gradient-type methods, because approximate second-order derivative information is employed. In deterministic unconstrained optimization, quasi-Newton methods update the iterates using

$$x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla f(x_k), \quad (2.5)$$

where the stepsize  $\alpha_k$  is usually determined by line search techniques, and  $B_k$  is a positive definite matrix that approximates the Hessian matrix of  $f(x)$  at iterate  $x_k$ . One widely-used updating strategy for  $B_k$  is the following BFGS formula [35]:

$$(\text{BFGS}) : \quad B_{k+1} = B_k + \frac{y_k y_k^\top}{s_k^\top y_k} - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k}, \quad (2.6)$$

where  $s_k := x_{k+1} - x_k$  and  $y_k := \nabla f(x_{k+1}) - \nabla f(x_k)$ . It is known that (2.6) preserves the positive-definiteness of sequence  $\{B_k\}$ . BFGS method and the limited memory BFGS method [26] demonstrate faster convergence speed than gradient methods both theoretically and numerically. Interested readers are referred to [35] for more details on quasi-Newton methods in deterministic settings.

In the stochastic settings, since the exact gradients of  $f$  are not available, the update formula (2.6) cannot guarantee that  $B_{k+1}$  is positive definite. To overcome this difficulty, Mokhtari and Ribeiro [29] proposed the following updating formula which preserves the positive-definiteness of  $B_k$ :

$$x_{k+1} = x_k - \alpha_k (B_k^{-1} + \zeta_k I) G_k, \quad (2.7)$$

where  $\zeta_k$  is a safeguard parameter such that  $B_k^{-1} + \zeta_k I$  is uniformly positive definite for all  $k$ , and  $G_k$  is defined as

$$G_k = \frac{1}{m_k} \sum_{i=1}^{m_k} G(x_k, \xi_{k,i}), \quad (2.8)$$

where the positive integer  $m_k$  denotes the batch size in gradient samplings. From AS.2 we know that  $G_k$  has the following properties:

$$\mathbb{E}[G_k | x_k] = \nabla f(x_k), \quad \mathbb{E}[\|G_k - \nabla f(x_k)\|^2 | x_k] \leq \frac{\sigma^2}{m_k}. \quad (2.9)$$

We also make the following bound assumption on  $B_k$ . Note that similar assumption was required in [29].

**AS.3** There exist two positive scalars  $m$  and  $M$  such that

$$mI \preceq B_k^{-1} + \zeta_k I \preceq MI, \quad \text{for any } k,$$

where  $m$  and  $M$  are positive scalars.

From (2.8), it follows that  $G_k$  depends on random variables  $\xi_{k,1}, \dots, \xi_{k,m_k}$ . We denote  $\xi_k := (\xi_{k,1}, \dots, \xi_{k,m_k})$ . We use  $\xi_{[k]}$  to denote the collection of all the random variables in the first  $k$  iterations, i.e.,  $\xi_{[k]} := (\xi_1, \dots, \xi_k)$ . It is easy to see from (2.8) and (2.7) that the random variable  $x_{k+1}$  depends on  $\xi_{[k]}$  only. Since  $B_k$  depends on  $x_k$ , we make the following assumption on  $B_k$  ( $k \geq 2$ ) (note that  $B_1$  is pre-given in the initial setting):

**AS.4** For any  $k \geq 2$ , the random variable  $B_k$  depends only on  $\xi_{[k-1]}$ .

The following equality follows directly from **AS.4** and (2.9):

$$\mathbb{E}[B_k^{-1} G_k | \xi_{[k-1]}] = B_k^{-1} \nabla f(x_k).$$

We will see later that this equality plays a key role in analyzing our stochastic quasi-Newton methods. Moreover, both assumptions **AS.3** and **AS.4** can be realized and we will propose two specific updating schemes of  $B_k$  that satisfy **AS.3-4** in Section 4.

We now present a general framework of stochastic quasi-Newton methods (SQN) for solving (1.1) in Algorithm 2.1.

---

**Algorithm 2.1 SQN: Stochastic quasi-Newton method for nonconvex stochastic optimization (1.1)**

---

**Input:** Given  $x_1 \in \mathbb{R}^n$ , a positive definite matrix  $B_1 \in \mathbb{R}^{n \times n}$ , batch sizes  $\{m_k\}_{k \geq 1}$ , safeguard parameters  $\{\zeta_k\}_{k \geq 1}$  and stepsizes  $\{\alpha_k\}_{k \geq 1}$  satisfying

$$\sum_{i=0}^{+\infty} \alpha_i = +\infty, \quad \sum_{i=0}^{+\infty} \alpha_i^2 < +\infty. \quad (2.10)$$

- 1: **for**  $k = 1, 2, \dots$  **do**
- 2:   Calculate  $G_k$  through (2.8), i.e.,

$$G_k = \frac{1}{m_k} \sum_{i=1}^{m_k} G(x_k, \xi_{k,i}).$$

- 3:   Calculate  $x_{k+1}$  through (2.7), i.e.,

$$x_{k+1} = x_k - \alpha_k (B_k^{-1} + \zeta_k I) G_k.$$

- 4:   Generate  $B_{k+1}$  that satisfies assumptions **AS.3** and **AS.4**.
  - 5: **end for**
- 

We now analyze the convergence of Algorithm 2.1. Note that if the sequence of iterates  $\{x_k\}$  generated by Algorithm 2.1 lies in a compact set, then it follows from **AS.1** that  $\{\nabla f(x_k)\}$  is bounded. Then there exists  $\bar{M} > 0$  such that

$$\|\nabla f(x_k)\| \leq \bar{M}. \quad (2.11)$$

The following lemma provides a descent property of the objective value of Algorithm 2.1.

**LEMMA 2.1.** *Assume that  $\{x_k\}$  is generated by Algorithm 2.1 and (2.11) and assumptions **AS.1-4** hold. Then the expectation of function value  $f(x_{k+1})$  conditioned on  $x_k$  satisfies*

$$\mathbb{E}[f(x_{k+1})|x_k] \leq f(x_k) - \alpha_k m \|\nabla f(x_k)\|^2 + \frac{1}{2} L \alpha_k^2 M^2 \left( \bar{M}^2 + \frac{\sigma^2}{m_k} \right), \quad (2.12)$$

where the conditioned expectation is taken with respect to  $\xi_k$ .

*Proof.* Using **AS.1**, **AS.3** and (2.7), we have

$$\begin{aligned} & f(x_{k+1}) \\ & \leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L}{2} \|x_{k+1} - x_k\|^2 \\ & = f(x_k) - \alpha_k \langle \nabla f(x_k), (B_k^{-1} + \zeta_k I) G_k \rangle + \frac{L}{2} \alpha_k^2 \|(B_k^{-1} + \zeta_k I) G_k\|^2 \\ & \leq f(x_k) - \alpha_k \langle \nabla f(x_k), (B_k^{-1} + \zeta_k I) \nabla f(x_k) \rangle - \alpha_k \langle \nabla f(x_k), (B_k^{-1} + \zeta_k I) \delta_k \rangle + \frac{L}{2} \alpha_k^2 M^2 \|G_k\|^2, \end{aligned} \quad (2.13)$$

where  $\delta_k = G_k - \nabla f(x_k)$ . Taking expectation on both sides of (2.13) conditioned on  $x_k$  with respect to  $\xi_k$

and noticing that  $\mathbb{E}[\delta_k|x_k] = 0$ , we obtain from **AS.4** that

$$\mathbb{E}[f(x_{k+1})|x_k] \leq f(x_k) - \alpha_k \langle \nabla f(x_k), (B_k^{-1} + \zeta_k I) \nabla f(x_k) \rangle + \frac{L}{2} \alpha_k^2 M^2 \mathbb{E}[\|G_k\|^2|x_k]. \quad (2.14)$$

From (2.9), (2.11) and  $\mathbb{E}[\delta_k|x_k] = 0$ , we have the following relations:

$$\begin{aligned} \mathbb{E}[\|G_k\|^2|x_k] &= \mathbb{E}[\|G_k - \nabla f(x_k) + \nabla f(x_k)\|^2|x_k] \\ &= \mathbb{E}[\|\nabla f(x_k)\|^2|x_k] + \mathbb{E}[\|G_k - \nabla f(x_k)\|^2|x_k] + 2\mathbb{E}[\delta_k, \nabla f(x_k)|x_k] \\ &= \|\nabla f(x_k)\|^2 + \mathbb{E}[\|G_k - \nabla f(x_k)\|^2|x_k] \\ &\leq \bar{M}^2 + \frac{\sigma^2}{m_k}, \end{aligned}$$

which together with (2.14) and **AS.3** yields (2.12).  $\square$

Before proceeding our analysis, we introduce the definition of *supermartingale* (see [11] for more details).

**DEFINITION 2.1.** *Let  $\mathcal{F}_k$  be an increasing sequence of  $\sigma$ -algebra. If  $\{X_k\}$  is a stochastic process satisfying*

- (i)  $\mathbb{E}[|X_k|] < \infty$ ;
- (ii)  $X_k \in \mathcal{F}_k$  for all  $k$ ;
- (iii)  $\mathbb{E}[X_{k+1}|\mathcal{F}_k] \leq X_k$  for all  $k$ ,

*then  $\{X_k\}$  is said to be a supermartingale.*

The following theorem states the convergence of a nonnegative supermartingale (see, e.g., Theorem 5.2.9 in [11]).

**PROPOSITION 2.1.** *If  $\{X_k\}$  is a nonnegative supermartingale, then  $\lim_{k \rightarrow \infty} X_k \rightarrow X$  almost surely and  $\mathbb{E}[X] \leq \mathbb{E}[X_0]$ .*

Now we are ready to give the main convergence result of our stochastic quasi-Newton method (Algorithm 2.1). Its proof essentially follows Theorem 1 in [29], but our assumptions here are relatively weaker.

**THEOREM 2.1.** *Assume that (2.11) and assumptions **AS.1-4** hold for  $\{x_k\}$  generated by Algorithm 2.1 with batch size  $m_k = \bar{m}$  for any  $k$ . Then*

$$\liminf_{k \rightarrow \infty} \|\nabla f(x_k)\| = 0, \quad \text{with probability 1.} \quad (2.15)$$

*Proof.* Define

$$\begin{aligned} \gamma_k &:= f(x_k) + \frac{LM^2(\bar{M}^2 + \sigma^2/\bar{m})}{2} \sum_{i=k}^{\infty} \alpha_i^2, \\ \beta_k &:= \alpha_k m \|\nabla f(x_k)\|^2. \end{aligned}$$

Let  $\mathcal{F}_k$  be the  $\sigma$ -algebra measuring  $\gamma_k$ ,  $\beta_k$  and  $x_k$ . Then from (2.12) we have that

$$\begin{aligned} \mathbb{E}[\gamma_{k+1}|\mathcal{F}_k] &= \mathbb{E}[f(x_{k+1})|\mathcal{F}_k] + \frac{LM^2(\bar{M}^2 + \sigma^2/\bar{m})}{2} \sum_{i=k+1}^{\infty} \alpha_i^2 \\ &\leq f(x_k) - \alpha_k m \|\nabla f(x_k)\|^2 + \frac{LM^2(\bar{M}^2 + \sigma^2/\bar{m})}{2} \sum_{i=k}^{\infty} \alpha_i^2 \\ &= \gamma_k - \beta_k, \end{aligned} \quad (2.16)$$

which implies that

$$\mathbb{E}[\gamma_{k+1} - f^{low} | \mathcal{F}_k] \leq \gamma_k - f^{low} - \beta_k.$$

Since  $\beta_k \geq 0$ , we have  $0 \leq \mathbb{E}[\gamma_k - f^{low}] \leq \gamma_1 - f^{low} < +\infty$ . Then according to Definition 2.1,  $\{\gamma_k - f^{low}\}$  is a supermartingale. Therefore, Proposition 2.1 shows that there exists  $\gamma$  such that  $\lim_{k \rightarrow \infty} \gamma_k = \gamma$  with probability 1, and  $\mathbb{E}[\gamma] \leq \mathbb{E}[\gamma_1]$ . Note that from (2.16) we have  $\mathbb{E}[\beta_k] \leq \mathbb{E}[\gamma_k] - \mathbb{E}[\gamma_{k+1}]$ . Thus,

$$\mathbb{E}\left[\sum_{k=0}^{\infty} \beta_k\right] \leq \sum_{k=0}^{\infty} (\mathbb{E}[\gamma_k] - \mathbb{E}[\gamma_{k+1}]) < +\infty,$$

which further yields that

$$\sum_{k=0}^{\infty} \beta_k = m \sum_{k=0}^{\infty} \alpha_k \|\nabla f(x_k)\|^2 < +\infty \quad \text{with probability 1.}$$

Since  $\sum_{k=0}^{\infty} \alpha_k = +\infty$ , it follows that (2.15) holds.  $\square$

**REMARK 2.1.** *Note that in Algorithm 2.1 we require that the stepsizes  $\alpha_k$  satisfy (2.10). This condition is easy to be satisfied. For example, one very simple strategy is to set  $\alpha_k = O(1/k)$ . In the numerical experiments we will show later, we test the performance of the algorithm using different settings of  $\alpha_k$  that satisfy (2.10).*

**3. A general framework for randomized stochastic quasi-Newton method for (1.1).** In Section 2, we proposed a general framework for stochastic quasi-Newton methods and studied its convergence. In this section, we propose another algorithmic framework, which is called randomized stochastic quasi-Newton method (RSQN), for solving (1.1). RSQN is very similar to SQN (Algorithm 2.1), with the only difference being that RSQN returns the iterate from a randomly chosen iteration as the final approximate solution. The idea of returning the iterate from a randomly chosen iteration is inspired by the RSG method [17]. It is shown in [17] that by randomly choosing an iteration number  $R$ , RSG returns  $x_R$  as an  $\epsilon$ -solution, i.e.,  $\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \epsilon$  with the worst-case  $\mathcal{SFO}$ -calls complexity being  $O(\epsilon^{-2})$ . Inspired by RSG, we propose the following RSQN (Algorithm 3.1) and analyze its worst-case  $\mathcal{SFO}$ -calls complexity.

In the following, we give the worst-case  $\mathcal{SFO}$ -calls complexity of Algorithm 3.1 for returning  $x_R$  such that  $\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \epsilon$ .

**THEOREM 3.1.** *Assume assumptions **AS.1-4** hold, and the stepsizes  $\alpha_k$  in Algorithm 3.1 are chosen such that  $0 < \alpha_k \leq 2m/(LM^2)$  with  $\alpha_k < 2m/(LM^2)$  for at least one  $k$ . Moreover, suppose that the probability mass function  $P_R$  is given as follows:*

$$P_R(k) := \text{Prob}\{R = k\} = \frac{m\alpha_k - LM^2\alpha_k^2/2}{\sum_{k=1}^N (m\alpha_k - LM^2\alpha_k^2/2)}, \quad k = 1, \dots, N. \quad (3.1)$$

*Then for any  $N \geq 1$ , we have*

$$\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \frac{D_f + (LM^2\sigma^2)/2 \sum_{k=1}^N (\alpha_k^2/m_k)}{\sum_{k=1}^N (m\alpha_k - LM^2\alpha_k^2/2)}, \quad (3.2)$$

*where  $D_f := f(x_1) - f^{low}$  and the expectation is taken with respect to  $R$  and  $\xi_{[N]}$ .*



---

**Algorithm 3.1 RSQN: Randomized stochastic quasi-Newton method for nonconvex stochastic optimization (1.1)**


---

**Input:** Given maximum iteration number  $N$ ,  $x_1 \in \mathbb{R}^n$ , a positive definite matrix  $B_1 \in \mathbb{R}^{n \times n}$ , stepsizes  $\{\alpha_k\}_{k \geq 1}$ , batch sizes  $\{m_k\}_{k \geq 1}$  and positive safeguard parameters  $\{\zeta_k\}_{k \geq 1}$ . Randomly choose  $R$  according to probability mass function  $P_R$  supported on  $\{1, \dots, N\}$ .

**Output:**  $x_R$ .

- 1: **for**  $k = 1, 2, \dots, R$  **do**
- 2:   Calculate  $G_k$  through (2.8), i.e.,

$$G_k = \frac{1}{m_k} \sum_{i=1}^{m_k} G(x_k, \xi_{k,i}).$$

- 3:   Calculate  $x_{k+1}$  through (2.7), i.e.,

$$x_{k+1} = x_k - \alpha_k (B_k^{-1} + \zeta_k I) G_k.$$

- 4:   Generate  $B_{k+1}$  such that assumptions **AS.3** and **AS.4** hold.
  - 5: **end for**
- 

*Proof.* From (2.13) it follows that

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) - \alpha_k \langle \nabla f(x_k), (B_k^{-1} + \zeta_k I) \nabla f(x_k) \rangle - \alpha_k \langle \nabla f(x_k), (B_k^{-1} + \zeta_k I) \delta_k \rangle + \\ &\quad \frac{L}{2} \alpha_k^2 M^2 [\|\nabla f(x_k)\|^2 + 2 \langle \nabla f(x_k), \delta_k \rangle + \|\delta_k\|^2] \\ &\leq f(x_k) - \left( m \alpha_k - \frac{LM^2}{2} \alpha_k^2 \right) \|\nabla f(x_k)\|^2 + \frac{LM^2}{2} \alpha_k^2 \|\delta_k\|^2 + LM^2 \alpha_k^2 \langle \nabla f(x_k), \delta_k \rangle \\ &\quad - \alpha_k \langle \nabla f(x_k), (B_k^{-1} + \zeta_k I)^{-1} \delta_k \rangle, \end{aligned}$$

where  $\delta_k = G_k - \nabla f(x_k)$ . Summing up the above inequality over  $k = 1, \dots, N$  and noticing that  $\alpha_k \leq 2m/(LM^2)$ , we have

$$\begin{aligned} &\sum_{k=1}^N \left( m \alpha_k - \frac{LM^2}{2} \alpha_k^2 \right) \|\nabla f(x_k)\|^2 \\ &\leq f(x_1) - f^{low} + \frac{LM^2}{2} \sum_{k=1}^N \alpha_k^2 \|\delta_k\|^2 + \sum_{k=1}^N (LM^2 \alpha_k^2 \langle \nabla f(x_k), \delta_k \rangle - \alpha_k \langle \nabla f(x_k), (B_k^{-1} + \zeta_k I)^{-1} \delta_k \rangle). \end{aligned} \quad (3.3)$$

Notice that both  $x_k$  and  $B_k$  depend only on  $\xi_{[k-1]}$ . Thus, by **AS.2** and **AS.4** we have that

$$\mathbb{E}_{\xi_k} [\langle \nabla f(x_k), \delta_k \rangle | \xi_{[k-1]}] = 0, \quad \mathbb{E}_{\xi_k} [\langle \nabla f(x_k), (B_k^{-1} + \zeta_k I) \delta_k \rangle | \xi_{[k-1]}] = 0.$$

Moreover, from (2.9) it follows that  $\mathbb{E}_{\xi_k} [\|\delta_k\|^2 | \xi_{[k-1]}] \leq \sigma^2/m_k$ . Therefore, taking the expectation on both sides of (3.3) with respect to  $\xi_{[N]}$  yields

$$\sum_{k=1}^N (m \alpha_k - LM^2 \alpha_k^2 / 2) \mathbb{E}_{\xi_{[N]}} [\|\nabla f(x_k)\|^2] \leq f(x_1) - f^{low} + \frac{LM^2 \sigma^2}{2} \sum_{k=1}^N \frac{\alpha_k^2}{m_k}. \quad (3.4)$$

Since  $R$  is a random variable with probability mass function  $P_R$  given in (3.1), it follows that

$$\mathbb{E}[\|\nabla f(x_R)\|^2] = \mathbb{E}_{R, \xi_{[N]}}[\|\nabla f(x_R)\|^2] = \frac{\sum_{k=1}^N (m\alpha_k - LM^2\alpha_k^2/2) \mathbb{E}_{\xi_{[N]}}[\|\nabla f(x_k)\|^2]}{\sum_{k=1}^N (m\alpha_k - LM^2\alpha_k^2/2)}, \quad (3.5)$$

which together with (3.4) implies (3.2).  $\square$

REMARK 3.1. *Different from SQN (Algorithm 2.1), stepsizes  $\alpha_k$  in RSQN (Algorithm 3.1) are not required to satisfy the condition (2.10). Besides, the assumption on the boundedness of  $\{\|\nabla f(x_k)\|\}$  is not needed in RSQN.*

The following complexity result follows immediately from Theorem 3.1.

COROLLARY 3.2. *Under the same conditions as in Theorem 3.1, further assume that the stepsizes  $\alpha_k = m/(LM^2)$  and the batch sizes  $m_k = \bar{m}$  for all  $k = 1, \dots, N$  for some integer  $\bar{m} \geq 1$ . Then the following holds*

$$\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \frac{2LM^2D_f}{Nm^2} + \frac{\sigma^2}{\bar{m}}, \quad (3.6)$$

where the expectation is taken with respect to  $R$  and  $\xi_{[N]}$ .

From Corollary 3.2 we can see that the right hand side of (3.6) depends on the batch size  $\bar{m}$ . Once  $\bar{m}$  is fixed, no matter how large the maximum iteration number  $N$  is, the right hand side of (3.6) is always lower bounded by  $\sigma^2/\bar{m}$ . Since we want  $\mathbb{E}[\|\nabla f(x_R)\|^2]$  to be as small as possible, we expect that it approaches zero when  $N$  is sufficiently large. Therefore,  $\bar{m}$  has to be chosen properly. The following corollary provides a choice of  $\bar{m}$  such that the worst-case  $\mathcal{SFO}$ -calls complexity of RSQN method is in the order of  $O(\epsilon^{-2})$  for obtaining an  $\epsilon$ -solution.

COROLLARY 3.3. *Let  $\bar{N}$  be the total number of  $\mathcal{SFO}$ -calls needed to calculate stochastic gradient  $G_k$  in Step 2 of Algorithm 3.1 for all the iterations. Under the same conditions as in Corollary 3.2, if we further assume that the batch size  $m_k$  is defined as*

$$m_k = \bar{m} := \left\lceil \min \left\{ \bar{N}, \max \left\{ 1, \frac{\sigma}{L} \sqrt{\frac{\bar{N}}{\bar{D}}} \right\} \right\} \right\rceil, \quad (3.7)$$

where  $\bar{D}$  is some problem-independent positive constant, then we have

$$\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \frac{4LM^2D_f}{Nm^2} \left( 1 + \frac{\sigma}{L} \sqrt{\frac{\bar{N}}{\bar{D}}} \right) + \max \left\{ \frac{\sigma^2}{\bar{N}}, \frac{\sigma L \sqrt{\bar{D}}}{\sqrt{\bar{N}}} \right\}, \quad (3.8)$$

where the expectation is taken with respect to  $R$  and  $\xi_{[N]}$ .

*Proof.* Note that the number of iterations of Algorithm 3.1 is at most  $N = \lceil \bar{N}/\bar{m} \rceil$ . Obviously,  $N \geq \bar{N}/(2\bar{m})$ . From Corollary 3.2 we have that

$$\begin{aligned} \mathbb{E}[\|\nabla f(x_R)\|^2] &\leq \frac{2LM^2D_f}{Nm^2} + \frac{\sigma^2}{\bar{m}} \leq \frac{4LM^2D_f}{\bar{N}m^2} \bar{m} + \frac{\sigma^2}{\bar{m}} \\ &\leq \frac{4LM^2D_f}{\bar{N}m^2} \left( 1 + \frac{\sigma}{L} \sqrt{\frac{\bar{N}}{\bar{D}}} \right) + \max \left\{ \frac{\sigma^2}{\bar{N}}, \frac{\sigma L \sqrt{\bar{D}}}{\sqrt{\bar{N}}} \right\}, \end{aligned} \quad (3.9)$$

which completes the proof.  $\square$

The following corollary follows immediately from Corollary 3.3.

**COROLLARY 3.4.** *Under the same conditions as Corollary 3.3, for any given  $\epsilon > 0$ , we further assume that the total number of  $\mathcal{SFO}$  calls  $\bar{N}$  to calculate  $G_k$  in Step 2 of Algorithm 3.1 satisfies*

$$\bar{N} \geq \max \left\{ \frac{C_1^2}{\epsilon^2} + \frac{4C_2}{\epsilon}, \frac{\sigma^2}{L^2 \tilde{D}} \right\} \quad (3.10)$$

where

$$C_1 = \frac{4\sigma M^2 D_f}{m^2 \sqrt{\tilde{D}}} + \sigma L \sqrt{\tilde{D}}, \quad C_2 = \frac{4LM^2 D_f}{m^2},$$

and  $\tilde{D}$  is same as in (3.7). Then we have

$$\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \epsilon,$$

where the expectation is taken with respect to  $R$  and  $\xi_{[N]}$ . It follows that to achieve  $\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \epsilon$ , the number of  $\mathcal{SFO}$ -calls needed to compute  $G_k$  in Step 2 of Algorithm 3.1 is at most in the order of  $O(\epsilon^{-2})$ .

*Proof.* (3.10) indicates that

$$\sqrt{\bar{N}} \geq \frac{\sqrt{C_1^2 + 4\epsilon C_2}}{\epsilon} \geq \frac{\sqrt{C_1^2 + 4\epsilon C_2} + C_1}{2\epsilon}.$$

(3.10) also implies that  $\sigma^2/\bar{N} \leq \sigma L \sqrt{\tilde{D}}/\sqrt{\bar{N}}$ . Then from Corollary 3.3 we have that

$$\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \frac{4LM^2 D_f}{\bar{N} m^2} \left( 1 + \frac{\sigma}{L} \sqrt{\frac{\bar{N}}{\tilde{D}}} \right) + \frac{\sigma L \sqrt{\tilde{D}}}{\sqrt{\bar{N}}} = \frac{C_1}{\sqrt{\bar{N}}} + \frac{C_2}{\bar{N}} \leq \epsilon.$$

□

**REMARK 3.2.** *In Corollaries 3.3 and 3.4 we did not consider the  $\mathcal{SFO}$ -calls that may be involved in updating  $B_{k+1}$  in Step 4 of the algorithms. In the next section, we will consider two specific updating schemes for  $B_k$ , and analyze their  $\mathcal{SFO}$ -calls complexities for calculating  $B_k$ .*

**4. Two specific updating schemes for  $B_k$ .** In Sections 2 and 3, we proposed two general frameworks for stochastic quasi-Newton methods for solving (1.1) and analyzed their convergence and worst-case  $\mathcal{SFO}$ -calls complexity, respectively. In both frameworks, we require that the Hessian approximation  $B_k$  satisfies assumptions **AS.3** and **AS.4**. In this section, we study two specific updating schemes for  $B_k$  such that **AS.3** and **AS.4** always hold.

**4.1. Stochastic damped BFGS updating formula.** In the setting of deterministic optimization, the classical BFGS algorithm updates the  $B_k$  through the formula (2.6). It can be proved that  $B_{k+1}$  is positive definite as long as  $B_k$  is positive definite and  $s_k^\top y_k > 0$  (see, e.g., [35, 43]). Line search techniques are usually used to ensure that  $s_k^\top y_k > 0$  is satisfied. However, in stochastic quasi-Newton method, line search techniques cannot be used because the objective function value is assumed to be difficult to obtain. As a result, how to preserve the positive definiteness of  $B_k$  is a main issue in designing stochastic quasi-Newton algorithms.

In [29], the RES algorithm is proposed for strongly convex stochastic optimization, in which iterates are updated via (2.7) where  $\zeta_k$  is set as a positive constant  $\Gamma$ . The following formula is adopted in [29] for

calculating the difference of the gradients:

$$\hat{y}_k = \bar{G}_{k+1} - G_k - \hat{\delta}s_k,$$

where  $\hat{\delta} > 0$  and

$$\bar{G}_{k+1} := \frac{1}{m_k} \sum_{i=1}^{m_k} G(x_{k+1}, \xi_{k,i}).$$

It should be noted that the same sample set  $\{\xi_{k,1}, \dots, \xi_{k,m_k}\}$  is used to compute  $G_k$  and  $\bar{G}_{k+1}$ .  $B_{k+1}$  is then calculated by the shifted BFGS update:

$$B_{k+1} = B_k + \frac{\hat{y}_k \hat{y}_k^\top}{s_k^\top \hat{y}_k} - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \hat{\delta}I, \quad (4.1)$$

where the shifting term  $\hat{\delta}I$  is added to prevent  $B_{k+1}$  from being close to singular. It is proved in [29] that  $B_{k+1} \succeq \hat{\delta}I$  under the assumption that  $f$  is strongly convex. However, (4.1) cannot guarantee the positive definiteness of  $B_{k+1}$  for nonconvex problems. Hence, we propose the following stochastic damped BFGS updating procedure (Procedure 4.1) for nonconvex problems. The damped BFGS updating procedure has been used in sequential quadratic programming method for constrained optimization in deterministic setting (see, e.g., [35]).

---

**Procedure 4.1 Stochastic Damped-BFGS update (SDBFGS)**

---

**Input:** Given  $\delta > 0$ ,  $\xi_k$ ,  $B_k$ ,  $G_k$ ,  $x_k$  and  $x_{k+1}$ .

**Output:**  $B_{k+1}$ .

- 1: Calculate  $s_k = x_{k+1} - x_k$  and calculate  $\hat{y}_k$  through

$$\hat{y}_k = \bar{G}_{k+1} - G_k - \delta s_k,$$

where  $\bar{G}_{k+1} := \frac{1}{m_k} \sum_{i=1}^{m_k} G(x_{k+1}, \xi_{k,i})$ .

- 2: Calculate

$$\hat{r}_k = \hat{\theta}_k \hat{y}_k + (1 - \hat{\theta}_k) B_k s_k,$$

where  $\hat{\theta}_k$  is calculated through:

$$\hat{\theta}_k = \begin{cases} 1, & \text{if } s_k^\top \hat{y}_k \geq 0.2 s_k^\top B_k s_k, \\ (0.8 s_k^\top B_k s_k) / (s_k^\top B_k s_k - s_k^\top \hat{y}_k), & \text{if } s_k^\top \hat{y}_k < 0.2 s_k^\top B_k s_k. \end{cases}$$

- 3: Calculate  $B_{k+1}$  through

$$B_{k+1} = B_k + \frac{\hat{r}_k \hat{r}_k^\top}{s_k^\top \hat{r}_k} - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \delta I. \quad (4.2)$$


---

REMARK 4.1. Notice that the most significant difference between Procedure 4.1 and RES lies in that  $\hat{r}_k$ , which is a convex combination of  $\hat{y}_k$  and  $B_k s_k$ , is used to replace  $\hat{y}_k$  in the updating formula (4.2) for  $B_{k+1}$ .

The following lemma shows that  $\{B_k\}$  obtained by Procedure 4.1 is uniformly positive definite.

LEMMA 4.1. *Suppose that  $B_k$  is positive definite, then  $B_{k+1}$  generated by Procedure 4.1 satisfies*

$$B_{k+1} \succeq \delta I. \quad (4.3)$$

*Proof.* From the definition of  $\hat{r}_k$ , we have that

$$s_k^\top \hat{r}_k = \hat{\theta}_k(s_k^\top \hat{y}_k - s_k^\top B_k s_k) + s_k^\top B_k s_k = \begin{cases} s_k^\top \hat{y}_k, & \text{if } s_k^\top \hat{y}_k \geq 0.2 s_k^\top B_k s_k, \\ 0.2 s_k^\top B_k s_k, & \text{if } s_k^\top \hat{y}_k < 0.2 s_k^\top B_k s_k, \end{cases}$$

which implies  $s_k^\top \hat{r}_k \geq 0.2 s_k^\top B_k s_k$ . Denote  $u_k = B_k^{\frac{1}{2}} s_k$ . Then we have

$$B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} = B_k^{\frac{1}{2}} \left( I - \frac{u_k u_k^\top}{u_k^\top u_k} \right) B_k^{\frac{1}{2}}.$$

Since  $I - \frac{u_k u_k^\top}{u_k^\top u_k} \succeq 0$  and  $s_k^\top \hat{r}_k > 0$ , we have that  $B_k + \frac{\hat{r}_k \hat{r}_k^\top}{s_k^\top \hat{r}_k} - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} \succeq 0$ . It then follows from (4.2) that  $B_{k+1} \succeq \delta I$ .  $\square$

From Lemma 4.1 we can see that, if starting with  $B_1 \succeq \delta I$ , we have  $B_k \succeq \delta I$  for all  $k$ . So if we further choose  $\zeta_k \geq \zeta$  for any positive constant  $\zeta$ , then it holds that

$$\zeta I \preceq B_k^{-1} + \zeta_k I \preceq \left( \frac{1}{\delta} + \zeta \right) I, \quad \text{for all } k,$$

which satisfies the assumption **AS.3** with  $m = \zeta$  and  $M = \zeta + 1/\delta$ . Moreover, Since  $\bar{G}_{k+1}$  is dependent only on  $\xi_k$ , it follows from (4.2) that  $B_{k+1}$  is dependent only on  $\xi_{[k]}$ , which satisfies the assumption **AS.4**. Therefore, we conclude that assumptions **AS.3** and **AS.4** hold for  $B_k$  generated by Procedure 4.1. We should also point out that in stochastic damped BFGS update Procedure 4.1, the shifting parameter  $\delta$  can be any positive scalar. But the shifting parameter  $\hat{\delta}$  in (4.1) used in RES is required to be smaller than the smallest eigenvalue of the Hessian of the strongly convex function  $f$ , which is usually negative for nonconvex problem.

Note that in Step 1 of Procedure 4.1, the stochastic gradient at  $x_{k+1}$  that is dependent on  $\xi_k$  is computed. Thus, when Procedure 4.1 is called at the  $k$ -th iteration to generate  $B_{k+1}$  in Step 4 of Algorithm 3.1, another  $m_k$   $\mathcal{SFO}$ -calls are needed. As a result, the number of  $\mathcal{SFO}$ -calls at the  $k$ -th iteration of Algorithm 3.1 becomes  $2m_k$ . This leads to the following complexity result for Algorithm 3.1.

THEOREM 4.1. *Denote  $N_{\mathcal{SFO}}$  as the total number of  $\mathcal{SFO}$ -calls in Algorithm 3.1 with Procedure 4.1 to generate  $B_{k+1}$ . Under the same conditions as in Corollary 3.4, to achieve  $\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \epsilon$ ,  $N_{\mathcal{SFO}}$  is at most  $2\bar{N}$  where  $\bar{N}$  satisfies (3.10), i.e., is in the order of  $O(\epsilon^{-2})$ .*

**4.2. Stochastic cyclic-BB-like updating formula.** Note that computing  $B_k^{-1} G_k$  in the updating formula for  $x_k$  (2.7) might be costly if  $B_k$  is dense or the problem dimension is large. To overcome this potential difficulty, we propose a cyclic Barzilai-Borwein (BB) like updating formula for  $B_k$  in this section. This updating formula can ensure that  $B_k$  is a diagonal matrix and thus very easy to be inverted.

The BB method has been studied extensively since it was firstly proposed in [1]. BB method is a gradient method with certain properties of quasi-Newton method. At the  $k$ -th iteration, the step size  $\alpha_k^{\text{BB}}$  for the

gradient method is calculated via

$$\alpha_k^{\text{BB}} := \arg \min_{\alpha \in \mathbb{R}} \|\alpha s_k - y_k\|^2, \quad \text{or} \quad \alpha_k^{\text{BB}} := \arg \min_{\alpha \in \mathbb{R}} \|s_k - y_k/\alpha\|^2,$$

where  $s_k := x_k - x_{k-1}$ ,  $y_k := \nabla f(x_k) - \nabla f(x_{k-1})$ . Direct calculations yield

$$\alpha_k^{\text{BB}} = \frac{s_k^\top y_k}{\|s_k\|^2}, \quad \text{or} \quad \alpha_k^{\text{BB}} = \frac{\|y_k\|^2}{s_k^\top y_k}.$$

Many studies have shown the superiority of BB methods over the classical gradient descent method in both theory and practical computation. Readers are referred to [20] for a relatively comprehensive discussion on BB methods. Besides, BB methods have been applied to solve many problems arising in real applications, such as image reconstruction [45, 34] and electronic structure calculation [46], and they have shown promising performance. Recently, the nice numerical behavior of cyclic BB (CBB) methods attracts a lot of attentions (see, e.g., [8, 20]). In CBB method, BB stepsize is used cyclicly, i.e., the stepsize in the  $l$ -th cycle is

$$\alpha_{ql+i} = \alpha_{q+1}^{\text{BB}}, \quad i = 1, \dots, q,$$

where  $q \geq 1$  is the cycle length and  $l = 0, 1, \dots$ . In the setting of deterministic optimization, line search techniques are usually adopted in CBB to ensure the global convergence. Although line search techniques are not applicable in stochastic optimization, we can still apply the idea of CBB to design an efficient algorithm that does not need to compute matrix inversion or solve linear equations in (2.7). The details of our procedure to generate  $B_k$  using stochastic CBB-like method are described as follows.

We set  $B_k := \lambda_k^{-1} I$ , and  $\lambda_k$  is updated as in CBB method  $\lambda_{ql+i} = \lambda_{q+1}^{\text{BB}}, i = 1, \dots, q$ , where  $q$  is the cycle length and  $l = 0, 1, \dots$ , and  $\lambda_{q+1}^{\text{BB}}$  is the optimal solution to

$$\min_{\lambda \in \mathbb{R}} \|\lambda^{-1} s_{ql} - y_{ql}\|^2, \quad \text{or} \quad \min_{\lambda \in \mathbb{R}} \|s_{ql} - \lambda y_{ql}\|^2, \quad (4.4)$$

where  $s_k = x_{k+1} - x_k$  and the gradient difference  $y_k$  is defined as

$$y_k = \bar{G}_{k+1} - G_k = \frac{\sum_{i=1}^{m_k} G(x_{k+1}, \xi_{k,i})}{m_k} - \frac{\sum_{i=1}^{m_k} G(x_k, \xi_{k,i})}{m_k}. \quad (4.5)$$

Direct calculations yield that  $\lambda_{q+1}^{\text{BB}} = s_{ql}^\top y_{ql} / \|y_{ql}\|^2$  or  $\lambda_{q+1}^{\text{BB}} = \|s_{ql}\|^2 / s_{ql}^\top y_{ql}$ . However,  $\lambda_{q+1}^{\text{BB}}$  calculated in this way might be negative since  $s_{ql}^\top y_{ql} < 0$  might happen. Therefore, we must adapt the stepsize in order to preserve the positive definiteness of  $B_k$ . We thus propose the following strategy for calculating  $\lambda_k$ :

$$\lambda_{k+1} = \begin{cases} \lambda_k, & \text{if } \text{mod}(k, q) \neq 0, \\ 1, & \text{if } \text{mod}(k, q) = 0, s_k^\top y_k \leq 0 \\ P_{[\lambda_{\min}, \lambda_{\max}]} \frac{s_{ql}^\top y_{ql}}{\|y_{ql}\|^2} \quad \text{or} \quad P_{[\lambda_{\min}, \lambda_{\max}]} \frac{\|s_{ql}\|^2}{s_{ql}^\top y_{ql}}, & \text{if } \text{mod}(k, q) = 0, s_k^\top y_k > 0, \end{cases} \quad (4.6)$$

where  $P_{[\lambda_{\min}, \lambda_{\max}]}$  denotes the projection onto the interval  $[\lambda_{\min}, \lambda_{\max}]$ , where  $\lambda_{\min}$  and  $\lambda_{\max}$  are given parameters. Note that we actually switch to gradient descent method (by setting  $\lambda_k = 1$ ) if  $s_k^\top y_k < 0$ . In our numerical tests later we will report the frequency of BB steps in this procedure. Notice that  $B_k$  generated

in this way satisfies the assumption **AS.3** with

$$m = \min\{\lambda_{\min}, 1\}, \quad M = \max\{\lambda_{\max}, 1\},$$

and in this case we can set  $\zeta_k = 0$  for all  $k$  in (2.7).

The stochastic CBB updating procedure for  $B_{k+1}$  is summarized formally in Procedure 4.2.

---

**Procedure 4.2 Stochastic Cyclic-BB-like update (SCBB)**

---

**Input:** Given  $q \in \mathbb{N}_+$ ,  $G_k, \lambda_{\min}, \lambda_{\max} \in \mathbb{R}^n$  with  $0 < \lambda_{\min} < \lambda_{\max}$ ,  $\xi_k, G_k, x_k$  and  $x_{k+1}$ .

**Output:**  $B_{k+1}$ .

1: **if**  $\text{mod}(k, q) = 0$  **then**

2:   Calculate  $s_k = x_{k+1} - x_k$  and

$$y_k = \frac{\sum_{i=1}^{m_k} G(x_{k+1}, \xi_{k,i})}{m_k} - G_k;$$

3:   **if**  $s_k^\top y_k > 0$  **then**

4:      $\lambda_{k+1} = P_{[\lambda_{\min}, \lambda_{\max}]} \frac{s_k^\top y_k}{\|y_k\|^2}$  or  $P_{[\lambda_{\min}, \lambda_{\max}]} \frac{\|s_k\|^2}{s_k^\top y_k}$ ;

5:   **else**

6:      $\lambda_{k+1} = 1$ ;

7:   **end if**

8: **else**

9:    $\lambda_{k+1} = \lambda_k$ ;

10: **end if**

11: Set  $B_{k+1} = \lambda_{k+1}^{-1} I$ .

---

When Procedure 4.2 is used to generate  $B_{k+1}$  in Step 4 of Algorithm 3.1, we have the following complexity result on  $\mathcal{SFO}$ -calls.

**THEOREM 4.2.** *Denote  $N_{sfo}$  as the total number of  $\mathcal{SFO}$ -calls in Algorithm 3.1 with Procedure 4.2 called to generate  $B_{k+1}$  at each iteration. Under the same conditions as Corollary 3.4, to achieve  $\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \epsilon$ ,  $N_{sfo}$  is at most  $\lceil (1+q)\bar{N}/q \rceil$  where  $\bar{N}$  satisfies (3.10), i.e.,  $N_{sfo}$  is in the order of  $O(\epsilon^{-2})$ .*

*Proof.* Under the same conditions as Corollary 3.4, the batch size  $m_k = \bar{m}$  for any  $k$ . If Procedure 4.2 is called at each iteration of Algorithm 3.1, then in every  $q$  iterations,  $\bar{m}(q+1)$   $\mathcal{SFO}$ -calls are needed. Since to achieve  $\mathbb{E}[\|\nabla f(x_R)\|^2] \leq \epsilon$  the number  $\mathcal{SFO}$  calls in Step 2 of Algorithm 3.1 is at most  $\bar{N}$ , the total number of  $\mathcal{SFO}$ -calls in Algorithm 3.1 is at most  $\lceil (1+q)\bar{N}/q \rceil$ .  $\square$

**5. Numerical Experiments.** In this section, we conduct numerical experiments to test the practical performance of the proposed algorithms.

By combining Algorithms 2.1 and 3.1 with Procedures 4.1 and 4.2, we get the following four algorithms: SDBFGS (Algorithm 2.1 with Procedure 4.1), SCBB (Algorithm 2.1 with Procedure 4.2), RSDBFGS (Algorithm 3.1 with Procedure 4.1), and RSCBB (Algorithm 3.1 with Procedure 4.2). We compare them with three existing methods for solving (1.1): SGD, RSG [17] and RES [29].

Since the course of these algorithms is a stochastic process, we run each instance  $N_{run}$  times and report the performance in average. In particular, we report the number of  $\mathcal{SFO}$ -calls ( $N_{sfo}$ ), the CPU time (in seconds), and the mean and variance (var.) of  $\|\nabla f(x_k^*)\|$  (or  $\|\nabla f(x_k^*)\|^2$ ) over  $N_{run}$  runs, where  $x_k^*$  is the output of the tested algorithm at  $k$ -th run with  $k = 1, \dots, N_{run}$ .

All the algorithms are implemented in Matlab R2013a on a PC with a 2.60 GHz Intel microprocessor

and 8GB of memory.

**5.1. A convex stochastic optimization problem.** We first consider a convex stochastic optimization problem, which is also considered in [29]:

$$\min_{x \in \mathbb{R}^n} f(x) = \mathbb{E}_\xi[f(x, \xi)] := \mathbb{E}\left[\frac{1}{2}x^\top(A + \text{Adiag}(\xi))x - b^\top x\right], \quad (5.1)$$

where  $\xi$  is uniformly drawn from  $\Xi := [-0.1, 0.1]^n$ ,  $b$  is chosen uniformly randomly from  $[0, 1]^n$ , and  $A$  is a diagonal matrix whose diagonal elements are uniformly chosen from a discrete set  $\mathcal{S}$  which will be specified later. We can control the condition number of (5.1) through the choice of  $\mathcal{S}$  and we will explore the performances of algorithms under different condition numbers.

For (5.1), we compare SDBFGS and SCBB with SGD and RES. For SGD, we tested two different choices of stepsize, i.e.,  $\alpha_k = 10^2/(10^3 + k)$  and  $10^4/(10^4 + k)$ . We also tested some other choices for  $\alpha_k$ , but the performance with these two are relatively better. The parameters for the other three algorithms are set as follows:

$$\begin{aligned} \text{SCBB: } \alpha_k &= \frac{10^2}{10^3 + k}, \quad \lambda_{\min} = 10^{-6}, \quad \lambda_{\max} = 10^8, \quad q = 5, \\ \text{SDBFGS: } \alpha_k &= \frac{10^2}{10^3 + k}, \quad \zeta_k = 10^{-4}, \quad \delta = 10^{-3}, \\ \text{RES: } \alpha_k &= \frac{10^2}{10^3 + k}, \quad \Gamma = 10^{-4}, \quad \hat{\delta} = 10^{-3}. \end{aligned}$$

Note that the parameter settings for RES are the same as the ones used in [29]. To make a fair comparison with RES, we thus adopted the same stepsize in these three algorithms above.

Since the solution of (5.1) is  $x^* = A^{-1}b$  if the random perturbation is ignored, we terminate the algorithms when

$$\frac{\|x_k - x^*\|}{\max\{1, \|x^*\|\}} \leq \rho,$$

where  $\rho > 0$  is a given tolerance. We chose  $\rho = 0.01$  in our experiments. We set the batch size  $m_k = 5$  for all the tested algorithms. Besides, for each instance the maximum iteration number is set as  $10^4$ . The results for different dimension  $n$  and set  $\mathcal{S}$  are reported in Table 5.1. Note that different choices of  $\mathcal{S}$  can reflect different condition numbers of (5.1).

From Table 5.1 we see that the performance of SGD is poor compared with the other three methods. The average number of  $\mathcal{SFO}$ -calls of SGD is significantly larger than the ones given by RES, SDBFGS and SCBB. Moreover, SGD diverges if the stepsize  $\alpha_k$  is too large or the condition number of the problem increases. It is also noticed that the performance of RES and SDBFGS is comparable. Furthermore, SCBB seems to be the best among the tested algorithms in terms of mean and variance of  $\|\nabla f(x_k^*)\|$  as well as the CPU time, although RES and SDBFGS need less number of  $\mathcal{SFO}$ -calls.

**5.2. A nonconvex support vector machine problem.** In this section, we compare RSDBFGS and RSCBB with RSG studied in [17] for solving the following nonconvex support vector machine problem with a sigmoid loss function (see [28])

$$\min_{x \in \mathbb{R}^n} f(x) := \mathbb{E}_{u,v}[1 - \tanh(v\langle x, u \rangle)] + \lambda \|x\|_2^2 \quad (5.2)$$



TABLE 5.1

Results for solving (5.1). Mean value and variance (var.) of  $\{\|\nabla f(x_k^*)\| : k = 1, \dots, N_{run}\}$  with  $N_{run} = 20$  are reported. “—” means that the algorithm is divergent.

$n$		SGD $\alpha_k = \frac{10^2}{10^3+k}$	SGD $\alpha_k = \frac{10^4}{10^4+k}$	RES $\alpha_k = \frac{10^2}{10^3+k}$	SDBFGS $\alpha_k = \frac{10^2}{10^3+k}$	SCBB $\alpha_k = \frac{10^2}{10^3+k}$
		$\mathcal{S} = \{0.1, 1\}$				
500	$N_{sfo}$	2.921e+03	2.400e+02	5.035e+02	5.025e+02	7.653e+02
	mean	9.781e-02	2.446e-01	9.933e-02	1.002e-01	1.123e-01
	var.	7.046e-07	1.639e-04	4.267e-06	7.329e-06	6.020e-06
	CPU	2.848e-01	2.580e-02	9.607e-01	6.273e-01	3.095e-02
1000	$N_{sfo}$	2.925e+03	2.380e+02	5.015e+02	5.000e+02	7.243e+02
	mean	1.453e-01	3.532e-01	1.476e-01	1.474e-01	1.667e-01
	var.	1.101e-06	1.368e-04	1.117e-05	8.251e-06	9.984e-06
	CPU	1.172e+00	8.665e-02	6.031e+00	6.109e+00	2.924e-01
5000	$N_{sfo}$	2.924e+03	2.400e+02	5.045e+02	5.045e+02	7.575e+02
	mean	3.165e-01	7.982e-01	3.194e-01	3.180e-01	3.624e-01
	var.	1.255e-06	2.050e-04	1.336e-05	1.246e-05	8.917e-06
	CPU	1.270e+01	7.661e-01	3.492e+02	3.577e+02	2.371e+00
		$\mathcal{S} = \{0.1, 1, 10\}$				
500	$N_{sfo}$	2.927e+03	5.000e+04	2.865e+02	2.875e+02	8.315e+03
	mean	1.622e-01	—	6.016e-01	5.698e-01	9.429e-02
	var.	5.421e-05	—	1.162e-02	9.589e-03	3.281e-06
	CPU	3.041e-01	4.842e+00	5.132e-01	3.683e-01	7.994e-01
1000	$N_{sfo}$	2.928e+03	5.000e+04	2.875e+02	2.880e+02	7.101e+03
	mean	2.137e-01	—	7.707e-01	7.791e-01	1.372e-01
	var.	4.638e-05	—	7.222e-03	6.860e-03	2.834e-06
	CPU	9.459e-01	1.934e+01	3.354e+00	3.595e+00	2.855e+00
5000	$N_{sfo}$	2.925e+03	5.000e+04	2.865e+02	2.865e+02	8.035e+03
	mean	4.911e-01	—	1.957e+00	1.956e+00	2.903e-01
	var.	6.575e-05	—	3.916e-02	4.517e-02	5.618e-06
	CPU	1.564e+01	1.525e+02	2.023e+02	2.039e+02	2.254e+01
		$\mathcal{S} = \{0.1, 1, 10, 100\}$				
500	$N_{sfo}$	5.000e+04	5.000e+04	6.279e+03	6.409e+03	4.953e+04
	mean	—	—	3.193e-01	3.479e-01	2.049e-01
	var.	—	—	6.003e-02	6.754e-02	9.889e-05
	CPU	3.136e+00	3.517e+00	8.458e+00	9.817e+00	3.955e+00
1000	$N_{sfo}$	5.000e+04	5.000e+04	9.028e+03	9.016e+03	5.644e+04
	mean	—	—	5.615e-01	5.005e-01	2.397e-01
	var.	—	—	6.704e-02	8.857e-02	2.132e-04
	CPU	1.774e+01	1.203e+01	1.251e+02	1.267e+02	1.169e+01
5000	$N_{sfo}$	5.000e+04	5.000e+04	6.756e+03	6.694e+03	6.000e+04
	mean	—	—	9.388e+00	1.104e+01	1.118e+00
	var.	—	—	4.133e+01	5.345e+01	2.581e-04
	CPU	1.534e+02	3.041e+03	3.022e+03	5.820e+03	1.278e+02

where  $\lambda > 0$  is a regularization parameter,  $u \in \mathbb{R}^n$  denotes the feature vector,  $v \in \{-1, 1\}$  refers to the corresponding label and  $(u, v)$  is drawn from the uniform distribution on  $[0, 1]^n \times \{-1, 1\}$ . Note that we do not compare with RES here because RES is designed for solving strongly convex problems. In order to compare with RSG, we adopt the same experimental settings as in [18]. The regularization parameter  $\lambda$  is set as 0.01. The initial point is set as  $x_1 = 5 * \bar{x}_1$ , where  $\bar{x}_1$  is drawn from the uniform distribution over  $[0, 1]^n$ . At the  $k$ -th iteration, to compute the stochastic gradient at iterate  $x_k$ , a sparse vector  $u_k$  with 5% nonzero components is first generated following the uniform distribution on  $[0, 1]^n$ , and then  $v_k$  is computed through  $v_k = \text{sign}(\langle \bar{x}, u_k \rangle)$  for some  $\bar{x} \in \mathbb{R}^n$  drawn from uniform distribution on  $[-1, 1]^n$ . Note that here the batch size  $m_k$  is equal to 1.

The code of RSG was downloaded from <http://www.ise.ufl.edu/glan/computer-codes>. In order to make a fair comparison, we generate our codes RSDBFGS and RSCBB by replacing the update formula (2.3) in RSG by SDBFGS and SCBB procedures. In both RSDBFGS and RSCBB, we adopt the same stepsize as in RSG. Note that an auxiliary routine is implemented to estimate the Lipschitz constant in [18]. The cycle

length in SCBB is set as  $q = 5$ . We test the three algorithms with different problem sizes  $n = 500, 1000$  and  $2000$  and different number of  $\mathcal{SFO}$ -calls  $N_{sfo} = 2500, 5000, 10000$  and  $20000$ . Recall that the theoretical performance of expectation of squared norm of gradient at returned point has been analyzed in Section 3. We next report the mean value and variance of  $\|\nabla f(x_k^*)\|^2$  over  $N_{run} = 20$  runs of each algorithm solving (5.2) in Table (5.2). To evaluate the quality of  $x_k^*$  in terms of classification, we also report the misclassification error on a testing set  $\{(u_i, v_i) : i = 1, \dots, K\}$ , which is defined as

$$\text{err}(x_k^*) := \frac{|\{i : v_i \neq \text{sign}(\langle x_k^*, u_i \rangle), i = 1, \dots, K\}|}{K},$$

and the sample size  $K = 75000$ . Here, the testing set is generated in the same way as we have introduced in previous paragraph.

From Table 5.2 we have the following observations. First, both RSDBFGS and RSCBB outperform RSG in terms of mean value and variance of  $\|\nabla f(x_k^*)\|^2$ , and in all cases RSDBFGS is the best. Second, both RSDBFGS and RSCBB outperform RSG in most cases in terms of misclassification error, and RSDBFGS is always the best. Third, RSDBFGS consumes most CPU time and RSG and RSCBB are comparable in terms of CPU time. Fourth, for fixed  $n$ , the misclassification error decreases when  $N_{sfo}$  increases.

Finally, we conduct some further tests to study the behavior of RSCBB. Note that in Procedure 4.2 we need to switch to a gradient step whenever  $s_k^T y_k < 0$  happens. So it is important to learn how often this happens in the course of the algorithm. In Table 5.3 we report the percentage of BB steps in RSCBB for solving (5.2). We can see from Table 5.3 that for fixed  $n$ , the percentage of BB steps monotonically decreases when  $N_{sfo}$  increases. Nonetheless, as we observed from the previous numerical tests, using BB steps helps significantly in improving the efficiency and accuracy of the algorithm.

**6. Conclusions and remarks.** In this paper we proposed two classes of stochastic quasi-Newton methods for nonconvex stochastic optimization. We first proposed a general framework of stochastic quasi-Newton methods, and analyzed its theoretical convergence in expectation. We further proposed a general framework of randomized stochastic quasi-Newton methods and established its worst-case  $\mathcal{SFO}$ -calls complexity. This kind of methods do not require the stepsize to converge to zero and provide an explicit worst-case  $\mathcal{SFO}$ -calls complexity bound. To create positive definite Hessian approximations that satisfy the assumptions required in the convergence and complexity analysis, we proposed two specific stochastic quasi-Newton update strategies, namely SDBFGS and SCBB strategies. We also studied their worst-case  $\mathcal{SFO}$ -calls complexities in the corresponding stochastic quasi-Newton algorithms. Finally, we reported some numerical experimental results that demonstrate the efficiency of our proposed algorithms. The numerical results indicate that the proposed SDBFGS and SCBB are preferable compared with some existing methods such as SGD, RSG and RES. We also noticed that the phenomenon shown in Table 5.3 deserves a further investigation to better understand the behavior of BB steps in designing stochastic quasi-Newton methods, and we leave this as a future work.

## REFERENCES

- [1] J. Barzilai and J.M. Borwein. Two point step size gradient methods. *IMA J. Numer. Anal.*, 8:141–148, 1988.
- [2] F. Bastin, C. Cirillo, and P. L. Toint. Convergence theory for nonconvex stochastic programming with an application to mixed logit. *Math. Program.*, 108:207–234, 2006.
- [3] A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful quasi-Newton stochastic gradient descent. *J. Mach. Learn. Res.*, 10:1737–1754, 2009.

TABLE 5.2

Results of RSG, RSDBFGS and RSCBB for solving (5.2). Mean value and variance (var.) of  $\{\|\nabla f(x_k^*)\|^2 : k = 1, \dots, N_{run}\}$  and average classification error (err.) with  $N_{run} = 20$  are reported.

$N_{sfo}$		RSG	RSDBFGS	RSCBB
$n = 500$				
2500	mean	3.622e-01	1.510e-02	3.021e-02
	var.	1.590e-01	3.041e-05	4.006e-04
	err.(%)	49.13	33.34	40.09
	CPU	3.794e+00	1.799e+01	4.136e+00
5000	mean	2.535e-01	1.441e-02	2.146e-02
	var.	1.320e-01	3.887e-05	3.464e-04
	err.(%)	45.47	31.09	36.37
	CPU	5.100e+00	3.452e+01	5.433e+00
10000	mean	3.201e-01	1.033e-02	6.277e-02
	var.	3.052e-01	1.770e-05	1.692e-02
	err.(%)	38.12	25.19	35.99
	CPU	7.127e+00	6.759e+01	6.969e+00
20000	mean	2.733e-01	1.117e-02	4.710e-02
	var.	2.923e-01	8.284e-05	6.327e-03
	err.(%)	30.81	24.59	31.60
	CPU	1.179e+01	1.227e+02	1.201e+01
$n = 1000$				
2500	mean	8.055e-01	1.966e-02	2.632e-02
	var.	4.250e-01	2.824e-05	1.087e-03
	err.(%)	48.39	37.07	45.32
	CPU	6.810e+00	8.172e+01	6.607e+00
5000	mean	5.188e-01	1.823e-02	3.088e-02
	var.	3.595e-01	1.717e-05	2.115e-04
	err.(%)	47.73	33.87	41.86
	CPU	8.059e+00	1.586e+02	8.296e+00
10000	mean	5.819e-01	1.744e-02	4.359e-02
	var.	8.432e-01	6.141e-05	1.543e-03
	err.(%)	45.44	30.61	40.07
	CPU	1.026e+01	3.890e+02	1.032e+01
20000	mean	4.457e-01	1.542e-02	5.062e-02
	var.	6.893e-01	3.920e-05	3.031e-03
	err.(%)	37.42	27.43	37.50
	CPU	1.841e+01	5.717e+02	1.673e+01
$n = 2000$				
2500	mean	2.731e+00	2.456e-02	5.366e-02
	var.	3.654e+00	1.019e-04	6.669e-03
	err.(%)	48.80	42.64	46.00
	CPU	1.219e+01	4.025e+02	1.246e+01
5000	mean	2.153e+00	2.076e-02	4.303e-02
	var.	3.799e+00	1.353e-04	1.211e-03
	err.(%)	48.54	40.97	44.43
	CPU	1.332e+01	1.048e+03	1.312e+01
10000	mean	8.216e-01	2.546e-02	3.756e-02
	var.	7.965e-01	1.084e-04	7.044e-04
	err.(%)	46.98	38.73	42.50
	CPU	1.840e+01	1.578e+03	1.716e+01
20000	mean	4.570e-01	2.199e-02	5.265e-02
	var.	3.782e-01	1.954e-04	2.820e-03
	err.(%)	44.00	33.81	40.94
	CPU	2.399e+01	3.215e+03	2.448e+01

TABLE 5.3

Percentage of BB steps in RSCBB for solving (5.2)

$n$	500				1000				2000			
$N_{sfo}$	2500	5000	10000	20000	2500	5000	10000	20000	2500	5000	10000	20000
Per.(%)	66.15	54.80	53.21	53.15	56.70	48.45	46.33	40.66	57.12	45.49	32.65	30.04

- [4] D. Brownstone, D. S. Bunch, and K. Train. Joint mixed logit models of stated and revealed preferences for alternative-fuel vehicles. *Transport. Res. B*, 34(5):315–338, 2000.
- [5] R. Byrd, G. Chin, W. Neveitt, and J. Nocedal. On the use of stochastic hessian information in optimization methods for machine learning. *SIAM J. Optim.*, 21(3):977–995, 2011.
- [6] R.H. Byrd, S.L. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-Newton method for large-scale optimization. *arXiv:1401.7020*, 2014.
- [7] K. L. Chung. On a stochastic approximation method. *Annals of Math. Stat.*, pages 463–483, 1954.
- [8] Y.H. Dai, W.W. Hager, K. Schittkowski, and H. Zhang. The cyclic Barzilai-Borwein method for unconstrained optimization. *IMA J. Numer. Anal.*, 26:604–627, 2006.
- [9] C. D. Dang and G. Lan. Stochastic block mirror descent methods for nonsmooth and stochastic optimization. Technical report, Department of Industrial and Systems Engineering, University of Florida, 2013.
- [10] J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 999999:2121–2159, 2011.
- [11] R. Durrett. *Probability: Theory and Examples*. Cambridge University Press, London, 2010.
- [12] Y. Ermoliev. Stochastic quasigradient methods and their application to system optimization. *Stochastics*, 9:1–36, 1983.
- [13] M. Fu. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14:192–215, 2002.
- [14] A. A. Gaivoronski. Nonstationary stochastic programming problems. *Kibernetika*, 4:89–92, 1978.
- [15] S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, i: a generic algorithmic framework. *SIAM J. Optim.*, 22:1469–1492, 2012.
- [16] S. Ghadimi and G. Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. Technical report, Department of Industrial and Systems Engineering, University of Florida, 2013.
- [17] S. Ghadimi and G. Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM J. Optim.*, 15(6):2341–2368, 2013.
- [18] S. Ghadimi and G. Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *Extended report*, <http://www.ise.ufl.edu/glan/files/2013/04/NonconvexSA-Techreport1.pdf>, 2013.
- [19] S. Ghadimi, G. Lan, and H. Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Math. Program.*, November, 2014.
- [20] W.W. Hager, B.A. Mair, and H. Zhang. An affine-scaling interior-point cbb method for box-constrained optimization. *Math. Program. Ser. A*, 119:1–32, 2009.
- [21] D. A. Hensher and W. H. Greene. The mixed logit model: The state of practice. *Transportation*, 30(2):133–176, 2003.
- [22] A. Juditsky, A. Nazin, A. B. Tsybakov, and N. Vayatis. Recursive aggregation of estimators via the mirror descent algorithm with average. *Probl. Inform. Transm.*, 41(4):78–96, 2005.
- [23] A. Juditsky, P. Rigollet, and A. B. Tsybakov. Learning by mirror averaging. *Annals of Stat.*, 36:2183–2206, 2008.
- [24] G. Lan. An optimal method for stochastic composite optimization. *Math. Program.*, 133(1):365–397, 2012.
- [25] G. Lan, A. S. Nemirovski, and A. Shapiro. Validation analysis of mirror descent stochastic approximation method. *Math. Program.*, 134:425–458, 2012.
- [26] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program., Ser. B*, 45(3):503–528, 1989.
- [27] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, 2009.
- [28] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent in function space. In *NIPS*, volume 12, pages 512–518, 1999.
- [29] A. Mokhtari and A. Ribeiro. Res: Regularized stochastic bfgs algorithm. *IEEE Trans. Signal Process.*, no. 10, 2014.
- [30] A. Mokhtari and A. Ribeiro. Global convergence of online limited memory bfgs. *arXiv:1409.2045v1 [math.OC]*, 6 Sep. 2014.
- [31] A. S. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. Optim.*, 19:1574–1609, 2009.
- [32] A. S. Nemirovski and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley-Interscience Series in Discrete Mathematics, John Wiley, 1983.
- [33] Y. E. Nesterov. *Introductory lectures on convex optimization: A basic course*. Applied Optimization. Kluwer Academic Publishers, Boston, MA, 2004.
- [34] T. Niu, X. Ye, Q. Fruhauf, M. Petrongolo, and L. Zhu. Accelerated barrier optimization compressed sensing (abocs) for ct reconstruction with improved convergence. *Phys. Med. Biol.*, 59:1801–1814, 2014.
- [35] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, USA, 2006.
- [36] B. T. Polyak. New stochastic approximation type procedures. *Automat. i Telemekh.*, 7:98–107, 1990.

- [37] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control and Optim.*, 30:838–855, 1992.
- [38] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Math. Stat.*, 22:400–407, 1951.
- [39] N.L. Roux and A.W. Fitzgibbon. A fast natural newton method. In *In Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 623–630, 2010.
- [40] A. Ruszczyński and W. Syski. A method of aggregate stochastic subgradients with on-line stepsize rules for convex stochastic programming problems. *Math. Prog. Stud.*, 28:113–131, 1986.
- [41] J. Sacks. Asymptotic distribution of stochastic approximation. *Annals of Math. Stat.*, 29:373–409, 1958.
- [42] N. N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-Newton method for online convex optimization. In *Proc. 11<sup>th</sup> Intl. Conf. Artificial Intelligence and Statistics (Aistats)*, pages 436–443, San Juan, Puerto Rico, 2007.
- [43] W. Sun and Y. Yuan. *Optimization theory and methods: nonlinear programming*. Springer Optimization and Its Applications, Vol. 1, 2006.
- [44] X. Wang, S. Ma, and Y. Yuan. Penalty methods with stochastic approximation for stochastic nonlinear programming. *arXiv:1312.2690*, 2013.
- [45] Y. Wang and S. Ma. Projected Barzilai-Borwein methods for large scale nonnegative image restorations. *Inverse Probl. Sci. En.*, 15(6):559–583, 2007.
- [46] X. Zhang, J. Zhu, Z. Wen, and A. Zhou. Gradient type optimization methods for electronic structure calculations. *SIAM J. Scientific Comput.*, 36(3):265–289, 2014.